BOUNDARY VALUE ANALYSIS

Why boundary value analysis?

"Bugs lurk in corners and congregate at boundaries."

- Errors frequently observed in the extreme ends of the input values.
- Programmers often fail to see the special processing required by the input values that lie at the boundaries.
- For example, programmers may improperly use **<** instead of **<=**, or conversely **<=** for **<**.

- BVA- also known as ‘range checking’.
- Another black box test case design technique.
- It is used to find the errors at boundaries of input domain and to create better test cases.
- We cannot test all the possible values because if done, the number of test cases would rise rapidly to the point of infeasibility!
  - For example; an Address text box which allows maximum 500 characters. So, writing test cases for each character once will be very difficult.

What is the critical assumption made with BVA?

- Single Fault Assumption
  - *Failures are rarely the result of simultaneous occurrence of two (or more) faults*

Based on this assumption

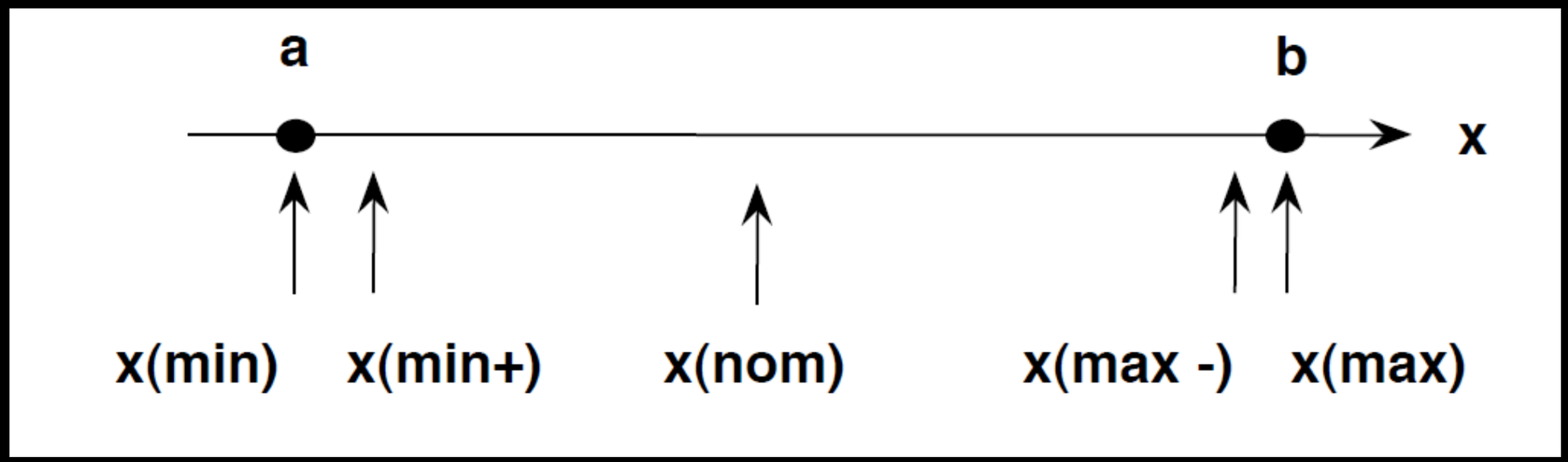
Number of Test Cases= 4*n +1

n= # of input variables

- Boundary testing is affected by this assumption is because this technique will create different testing scenarios by **focusing only the boundaries of a single data input**, disregarding the combination effect of multiple inputs.

The basic idea in boundary value testing is to select **input variable** values at their:

1. Minimum **(min)**
2. Just above the minimum **(min+1)**
3. A nominal value **(mid or nom)**
4. Just below the maximum **(max-1)**
5. Maximum **(max)**



Two-variable function test cases

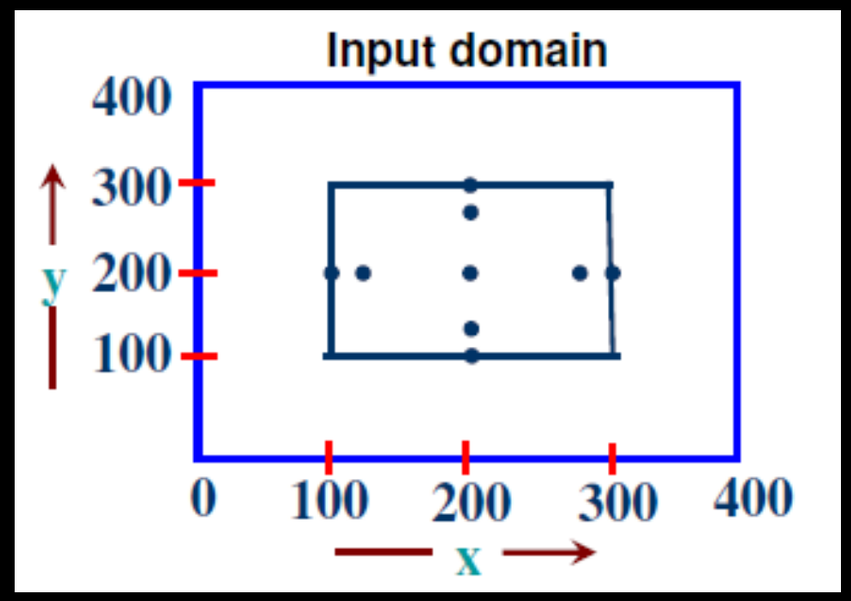
No. of test cases = 4*n +1 = 4*2 +1 = 9

$$F(x_1, x_2) : A \rightarrow B$$

$$a \leq x_1 \leq b \qquad c \leq x_2 \leq d$$

Redundant tests

$\langle x_{1nom}, x_{2min} \rangle$
$\langle x_{1nom}, x_{2min+} \rangle$
$\langle x_{1nom}, x_{2nom} \rangle$ ↔ $\langle x_{1min}, x_{2nom} \rangle$
$\langle x_{1min+}, x_{2nom} \rangle$
$\langle x_{1nom}, x_{2nom} \rangle$
$\langle x_{1nom}, x_{2max-} \rangle$
$\langle x_{1max-}, x_{2nom} \rangle$
$\langle x_{1nom}, x_{2max} \rangle$
$\langle x_{1max}, x_{2nom} \rangle$



Example 1

Consider a program that computes the intersection point of two straight lines and displays the result. It reads two integer pairs (m1, c1) and (m2, c2) defining the two straight lines of the form y=mx+ c.

The program output may have one of the following words: [Parallel lines, Intersecting lines, Coincident lines, Invalid input]

and the input values are from the interval [0,50] for c1,c2 and [0,10] for m1,m2.

Design the boundary value test cases.

Equation of a st. line is of the form

$$y=mx+c$$

Two straight lines will be:-

- Parallel lines if m1=m2, c1≠c2
- Intersecting lines if m1≠m2
- Coincident lines m1=m2, c1=c2

No. of input variables = n = 4 (m1, c1, m2, c2)

=> **No. of test cases = 4*n+1 = 4*4 + 1 = 17**

Boundary value test cases are:-

Test Case	m1	c1	m2	c2	Expected o/p
1	0	25	5	25	Intersecting
2	1	25	5	25	Intersecting
3	5	25	5	25	Coincident
4	9	25	5	25	Intersecting
5	10	25	5	25	Intersecting
6	5	0	5	25	Parallel
7	5	1	5	25	Parallel
8	5	49	5	25	Parallel
9	5	50	5	25	Parallel
10	5	25	0	25	Intersecting
11	5	25	1	25	Intersecting
12	5	25	9	25	Intersecting
13	5	25	10	25	Intersecting
14	5	25	5	0	Parallel
15	5	25	5	1	Parallel
16	5	25	5	49	Parallel
17	5	25	5	50	Parallel

Example 2

Consider a program which checks whether an input string is a palindrome or not.

The input string is of size 3.

The program output may have one of the following words:
[Palindrome, Not a palindrome],

and the input values belong to the domain [A-Z].

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define size 3
void main()
{
        char strsrc[size];
        char strtmp[size];
        clrscr();
        printf("\nEnter String:=");
        gets(strsrc);
        strcpy(strtmp,strupr(strsrc));
        strrev(strtmp);
        if(strcmp(strsrc,strtmp)==0)
        printf("\nEnter string %s is
pallindrom",strsrc);
        else
printf("\n Entered string %s,is not
pallindrome",strsrc);
        getch();
}
```

No. of test cases =4*n +1 = 4*3+1 = 13

The boundary value test cases are:-

Test case	Alphabet 1	Alphabet 2	Alphabet 3	Expected o/p
1	A	M	M	Not a palindrome
2	B	M	M	Not a palindrome
3	M	M	M	Not a palindrome
4	Y	M	M	Not a palindrome
5	Z	M	M	Not a palindrome
6	M	A	M	Not a palindrome
7	M	B	M	Not a palindrome
8	M	Y	M	Not a palindrome
9	M	Z	M	Not a palindrome
10	M	M	A	Not a palindrome
11	M	M	B	Not a palindrome
12	M	M	Y	Not a palindrome
13	M	M	Z	Not a palindrome

Limitations of BVA

Why we moved on to better testing techniques like Robustness, Worst case, etc?

- BVA works well when the Program Under Test (PUT) is a “function of several independent variables that represent bounded physical quantities”. When these conditions are met BVA works well but when they are not we can find deficiencies in the results.

- E.g.: PreviousDate problem, tester’s intuition and common sense shows that we require more emphasis towards the end of February or on leap years.

- The reason for this poor performance— BVA cannot compensate or take into consideration the nature of a function or the dependencies between its variables.

- It provides a relatively simple and formal testing. When issues arise such as dependencies between variables, BVA = restrictive

- Generally Boundary Value Testing techs. are computationally+theoretically inexpensive in the creation of test cases. For this reason, Boundary Value Analysis still has a part to play in modern day testing practices.